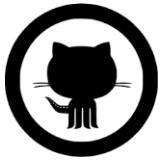
ROS. Моделирование роботов в среде Gazebo

Марков Алексей, ВолгГТУ

Нам понадобится



https://github.com/Garrus007/roboschool2018

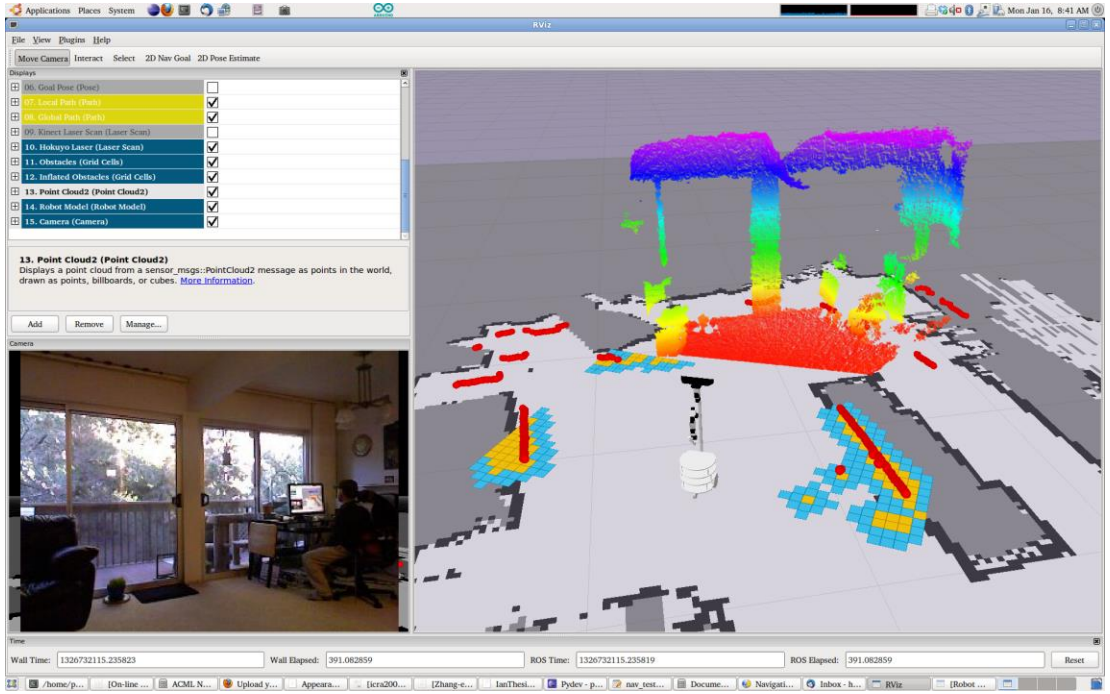
```
user@ros: ~/ros/src$ git clone https://github.com/Garrus007/roboschool2018
user@ros: ~/ros/src$ sudo apt install freerdp
user@ros: ~/ros/src$ sudo apt install ros-kinetic-teleop-twist-keyboard
user@ros: ~/ros/src$ echo “export
GAZEBO_MODEL_PATH=$(pwd)/roboschool2018/car_gazebo/models” >> ~/.bashrc
```

Что такое ROS?

ROS (Robot Operation System) – фреймворк для написания ПО для роботов. Это набор инструментов, библиотек и соглашений, которые упрощают задачу написания сложного ПО для разных робототехнических платформ.

Почему ROS?

- Упрощает написание модульного ПО
- Большое количество библиотек и инструментов
- Обширное коммьюнити
- Стандартизированные интерфейсы и легкая интеграция стороннего ПО

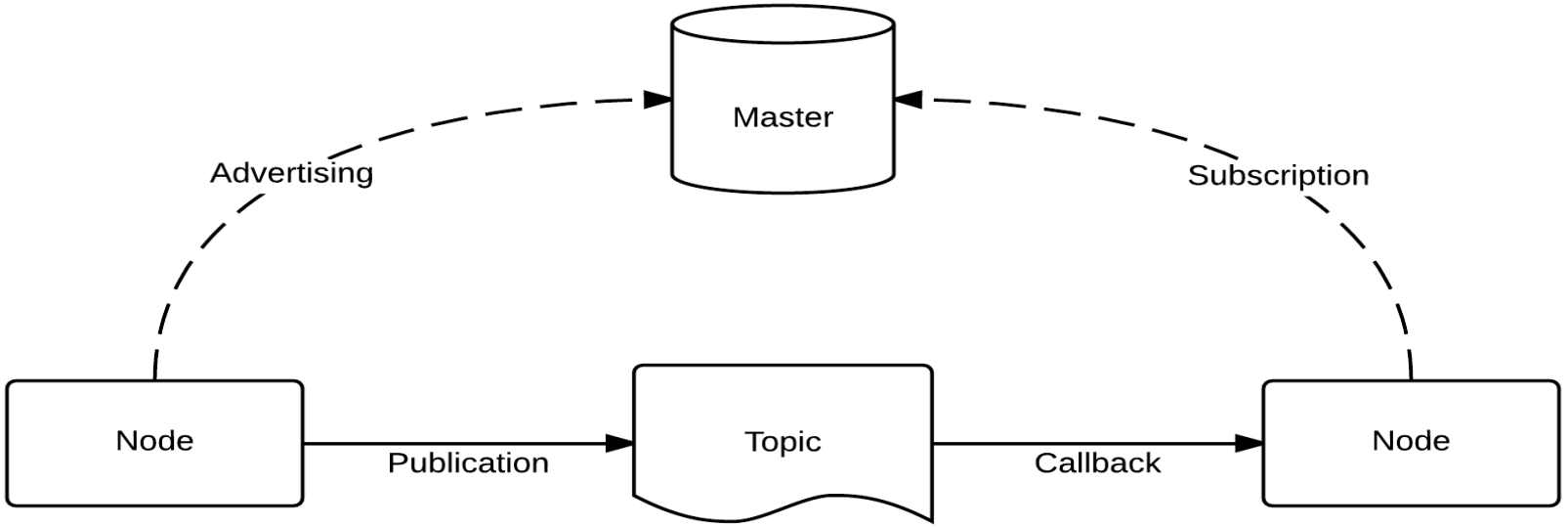


ROS rviz

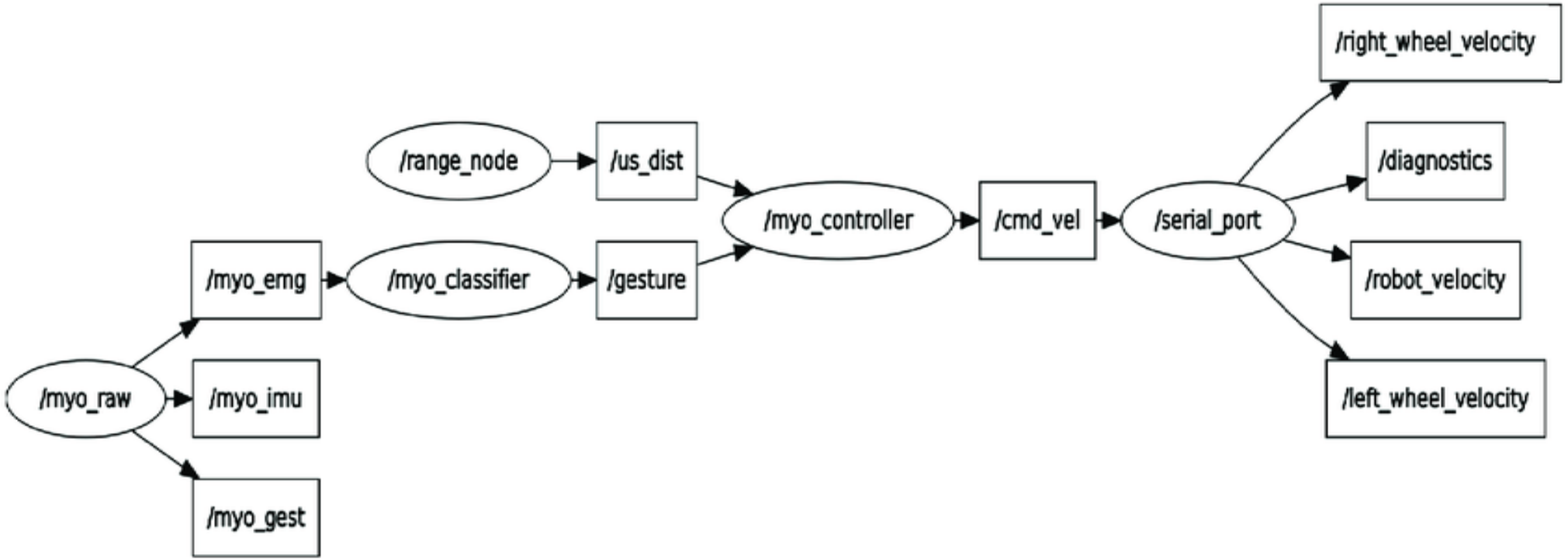
Архитектура с использованием ROS

- Программа состоит из отдельных процессов (nodes)
- Ноды взаимодействуют с помощью механизма топиков и сервисов
- ROS master — центральный процесс, осуществляющий механизм взаимодействия

Именование топиков:
/some/topic/name



Пример графа нод



Файловая структура

ros_workspace

\build

\ devel

\src

\your_package

Пример: publisher & subscriber

1. Создание пакета ROS

```
user@ros: ~/ros/src$ catkin_create_pkg beginner_tutorials std_msgs rospy
```

Название пакета

Зависимости

2. Создание исходников

```
user@ros: ~/ros/src$ cd beginner_tutorials
user@ros: ~/ros/src/beginner_tutorials$ mkdir scripts
user@ros: ~/ros/src/beginner_tutorials$ cd scripts
user@ros: ~/ros/src/beginner_tutorials$ touch publisher.py
user@ros: ~/ros/src/beginner_tutorials$ touch subscriber.py
user@ros: ~/ros/src/beginner_tutorials$ chmod +x publisher.py subscriber.py
```

Пример: publisher & subscriber

3. Написание publisher’a (publisher.py)

```
#!/usr/bin/env python
import rospy

if __name__ == '__main__':
    rospy.init_node('publisher')
    rospy.loginfo('Hello world')
```

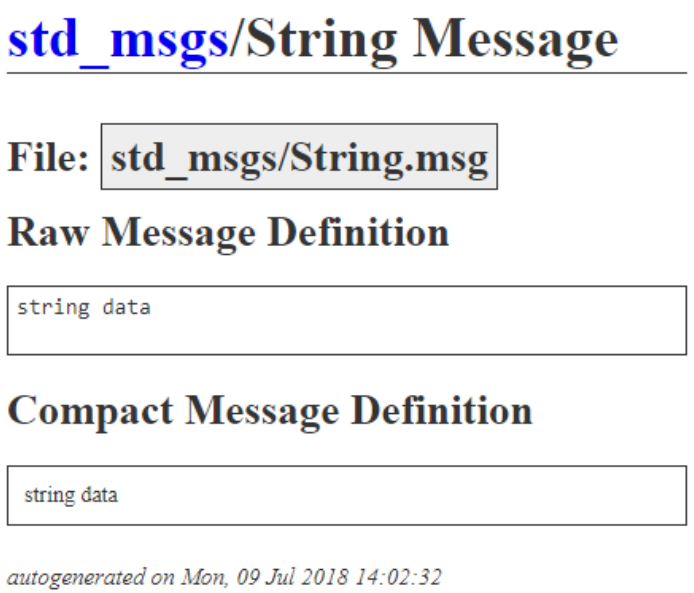
```
user@ros: ~/ros/$ roscore
```

```
user@ros: ~/ros/$ rosrun beginner_tutorials publisher.py
[INFO] [1542633272.254294]: Hello world
```

Пример: publisher & subscriber

3. Написание publisher’a (publisher.py)

Тип сообщения



std_msgs/String Message

File: std_msgs/String.msg

Raw Message Definition

```
string data
```

Compact Message Definition

```
string data
```

autogenerated on Mon, 09 Jul 2018 14:02:32

Пример: publisher & subscriber

3. Написание publisher'а (publisher.py)

```
#!/usr/bin/env python
#coding=utf-8
import rospy
from std_msgs.msg import String

if __name__ == '__main__':
    rospy.init_node('publisher')

    # Публишер - для публикации данных в
топик
    pub = rospy.Publisher('chatter',
String, queue_size=10)

# Для создания задержки (частота 5 Гц)
rate = rospy.Rate(5)

# Создаем сообщение
msg = String()
msg.data = 'Hello World'

# Публикуем
while not rospy.is_shutdown():
    pub.publish(msg)
    rate.sleep()
```

Пример: publisher & subscriber

4. Написание subscriber’a (subscriber.py)

```
#!/usr/bin/env python
#coding=utf-8

import rospy
from std_msgs.msg import String

def callback(msg):
        rospy.loginfo(msg.data)

if __name__ == '__main__':
        rospy.init_node('subscriber')
        rospy.Subscriber("chatter", String, callback)
        rospy.spin()
```

Пример: publisher & subscriber

5. Запуск

```
user@ros: ~/ros/$ roscore
```

```
user@ros: ~/ros/$ rosrun beginner_tutorials publisher.py
```

```
user@ros: ~/ros/$ rosrun beginner_tutorials subscriber.py
[INFO] [1542636601.987744]: Hello World
[INFO] [1542636602.187998]: Hello World
[INFO] [1542636602.387618]: Hello World
[INFO] [1542636602.587875]: Hello World
```

Пример: publisher & subscriber

6. Просмотр топиков

```
user@ros: ~/ros/$ rostopic list
/chatter
/rosout
/rosout_agg
```

7. Мониторинг топика

```
user@ros: ~/ros/$ rostopic echo /chatter
data: "Hello World"
---
data: "Hello World"
---
data: "Hello World"
---
```

Пример: publisher & subscriber

8. Граф

```
user@ros: ~/ros/$ rqt_graph
```

Gazebo

Gazebo – широко распространенный физический 3D симулятор роботов.

Возможности:

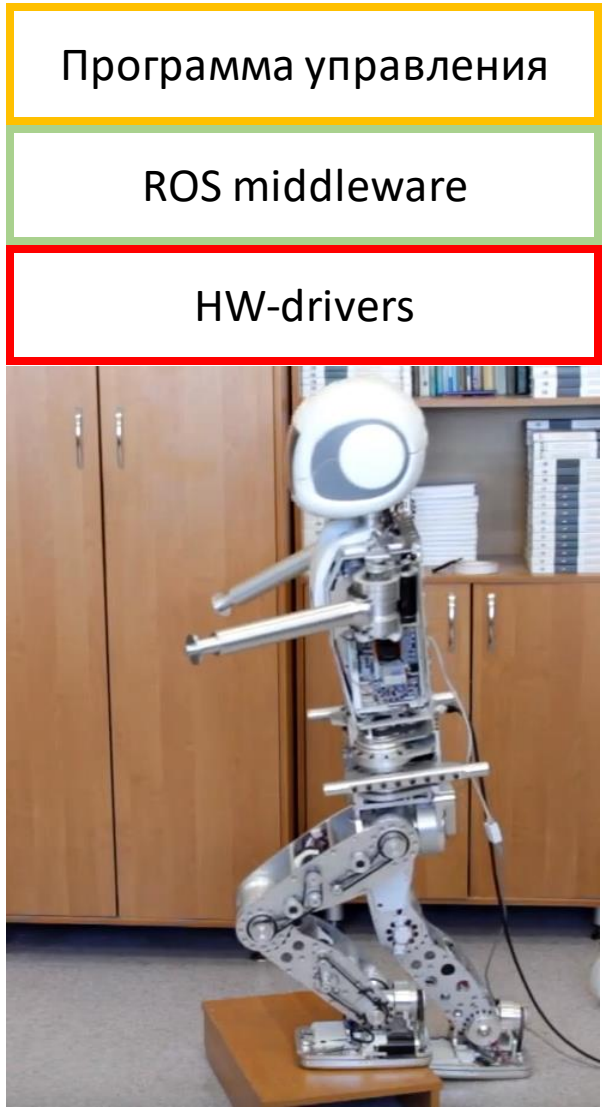
- Моделирование динамики
- Моделирование датчиков (камеры, лидары, Kinect, ИНС и другие)
- Богатая библиотека моделей роботов, предметов и окружения
- Интеграция с ROS

Почему Gazebo?

Возможность отладки алгоритмов управления робототехническими системами и их перенос на реальных роботов с минимальными изменениями.

Почему Gazebo?

Возможность отладки алгоритмов управления робототехническими системами и их перенос на реальных роботов с минимальными изменениями.



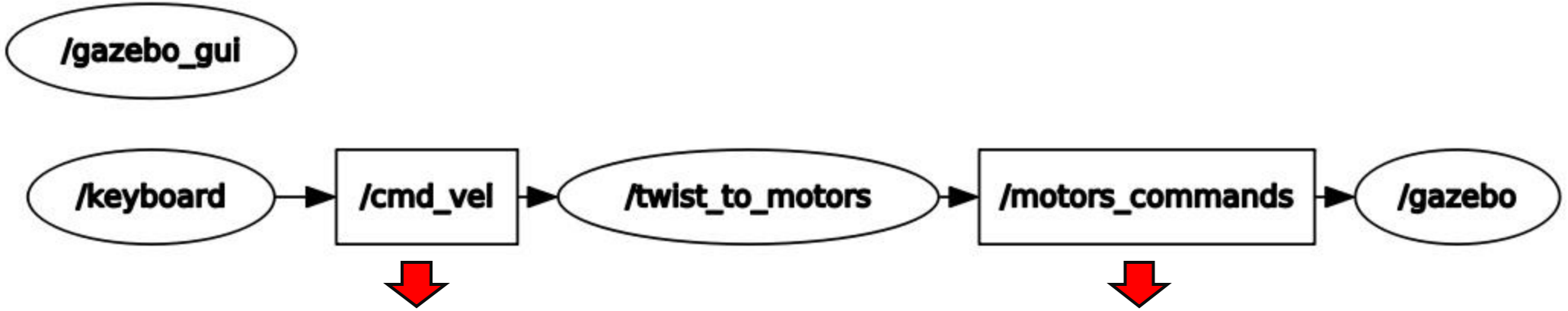
Пример: запуск Gazebo

```
user@ros: ~/ros/$ roslaunch car_gazebo keyboard.launch
```

```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
```

Пример: запуск Gazebo

```
user@ros: ~$ rqt_graph
```

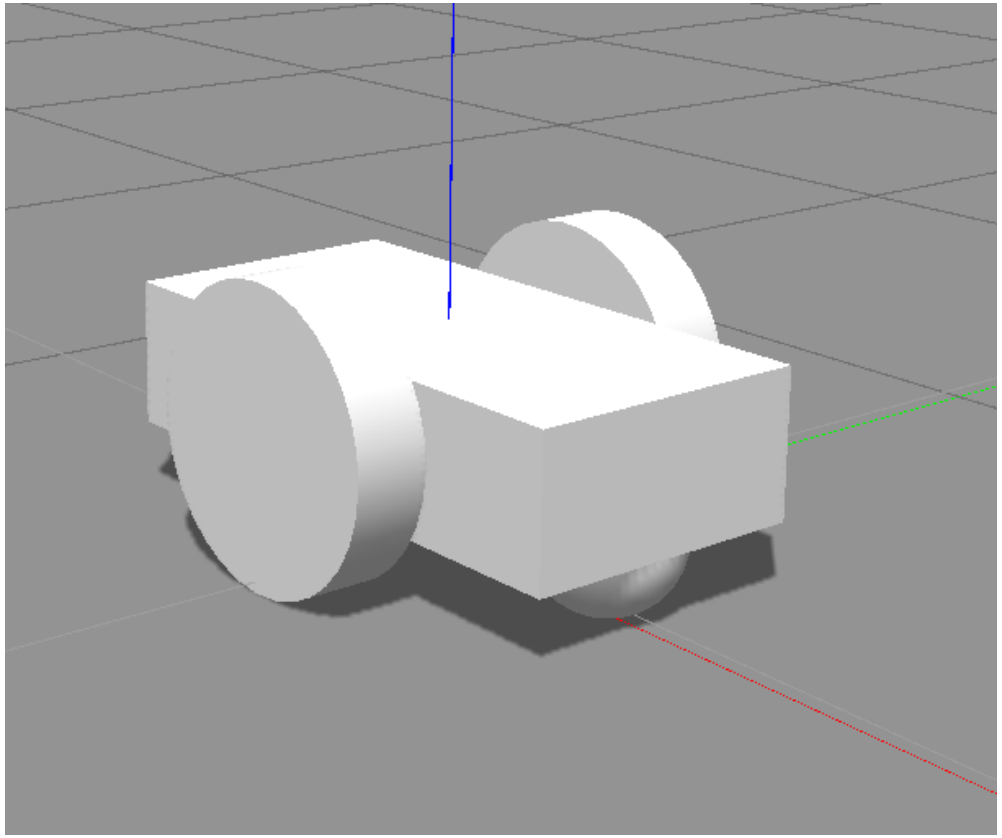



```
user@ros: ~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

```
user@ros: ~$ rosmsg show car_msgs/MotorsControl
int16 left
int16 right
```

Создание модели робота в Gazebo

Создадим двухколесного робота, которым можно управлять с клавиатуры



URDF vs SDF

URDF

- Де-факто стандарт ROS
- Описание только отдельных роботов
- Только масса и инерция
- Только геометрия, меш, цвет

SDF

- Формат $Gazebo^*$
- Новый расширенный формат
- Описание роботов, нескольких роботов, мира (aka карты)
- Большие физических свойств
- Больше визуальных свойств
- Расширяемый

http://sdformat.org

План

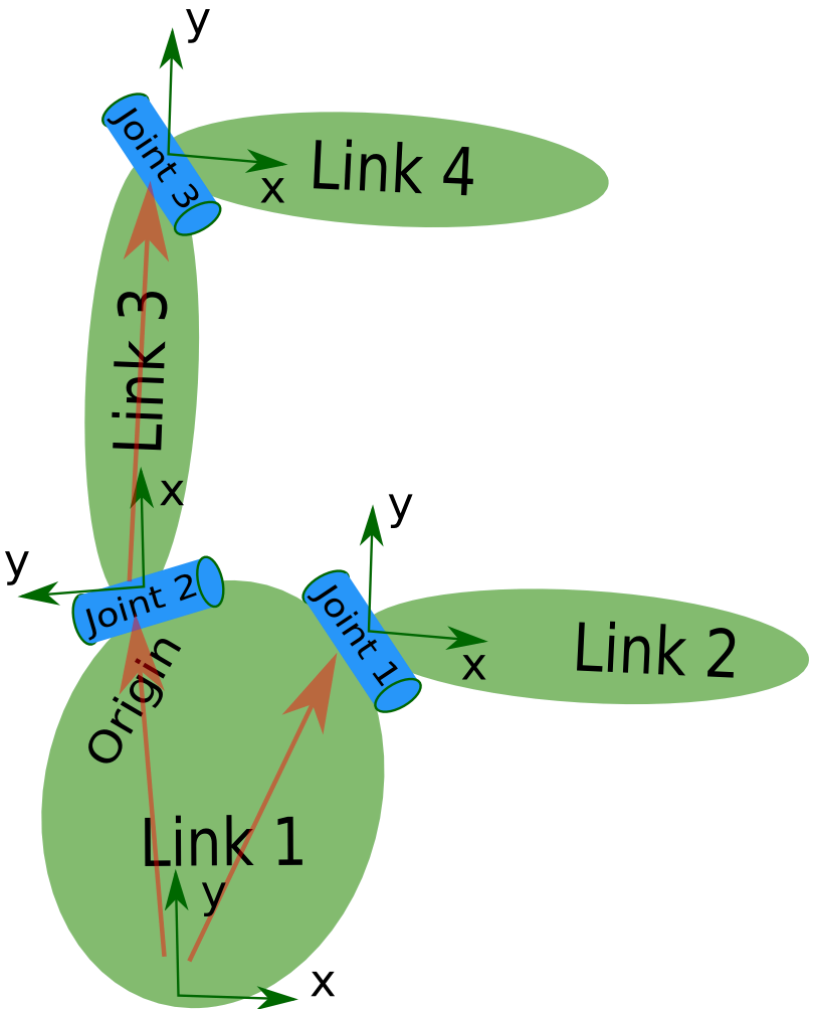
1. Описать структуру робота в URDF
    1. Описать тела и связи
    2. Описать внешний вид
    3. Описать коллизии
    4. Описать физические свойства
2. Создать плагин для Gazebo, чтобы управлять приводами

URDF

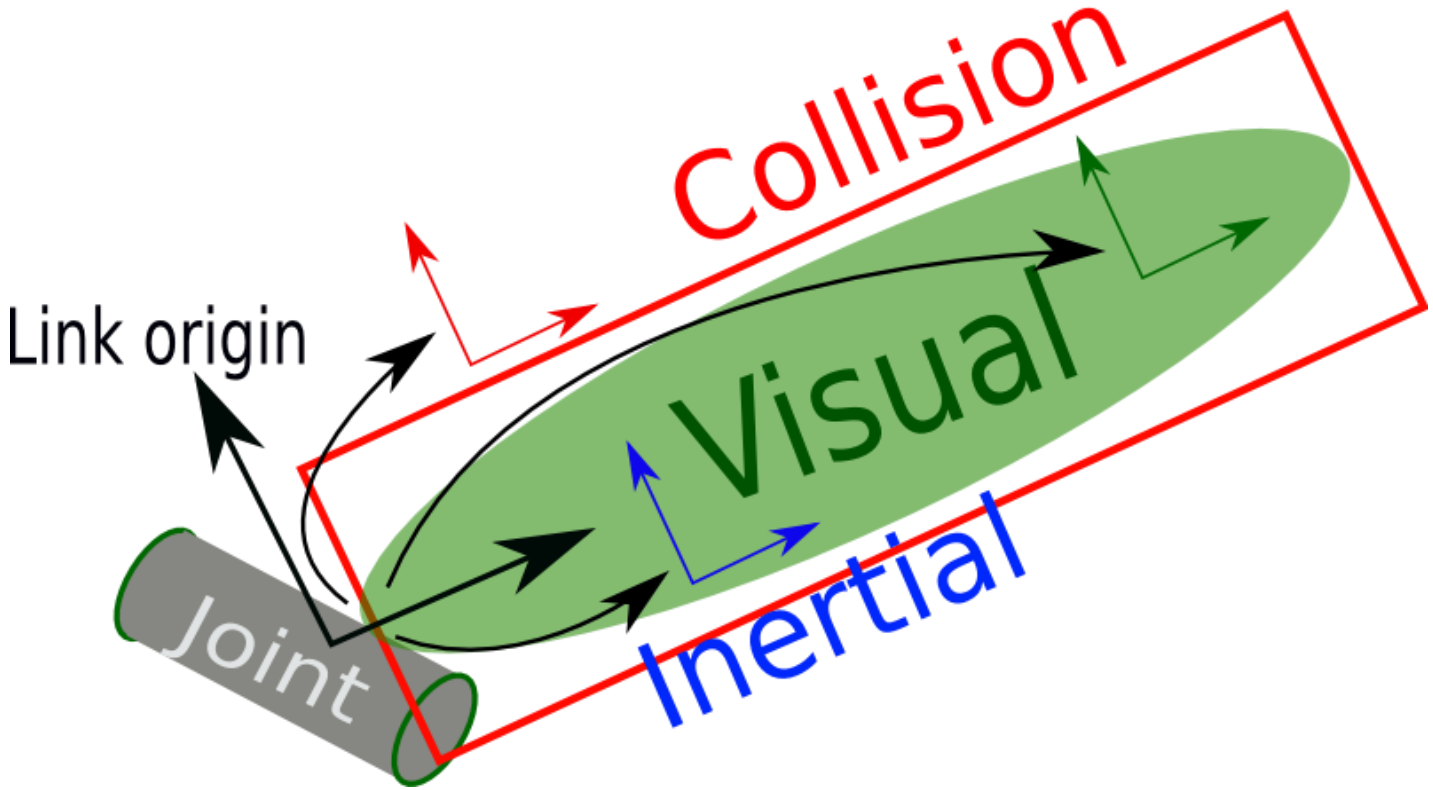
URDF (Unified Robot Description Format) – XML спецификация для описания роботов.

- Внешний вид
- Коллизии
- Физические свойства (масса, инерция)
- Связи различных типов

Xacro (XML Macros) – макро-язык XML, упрощает создание URDF



URDF Link



Создание проекта

Заготовка проекта уже сделана:

`roboschool2018/test_robot`

Создайте файлы :

- `urdf/test_robot.xacro` – главный файл
- `urdf/body.xacro` – описание корпуса
- `urdf/wheel.xacro` – описание колеса

1. Создание корпуса

test_robot.xacro

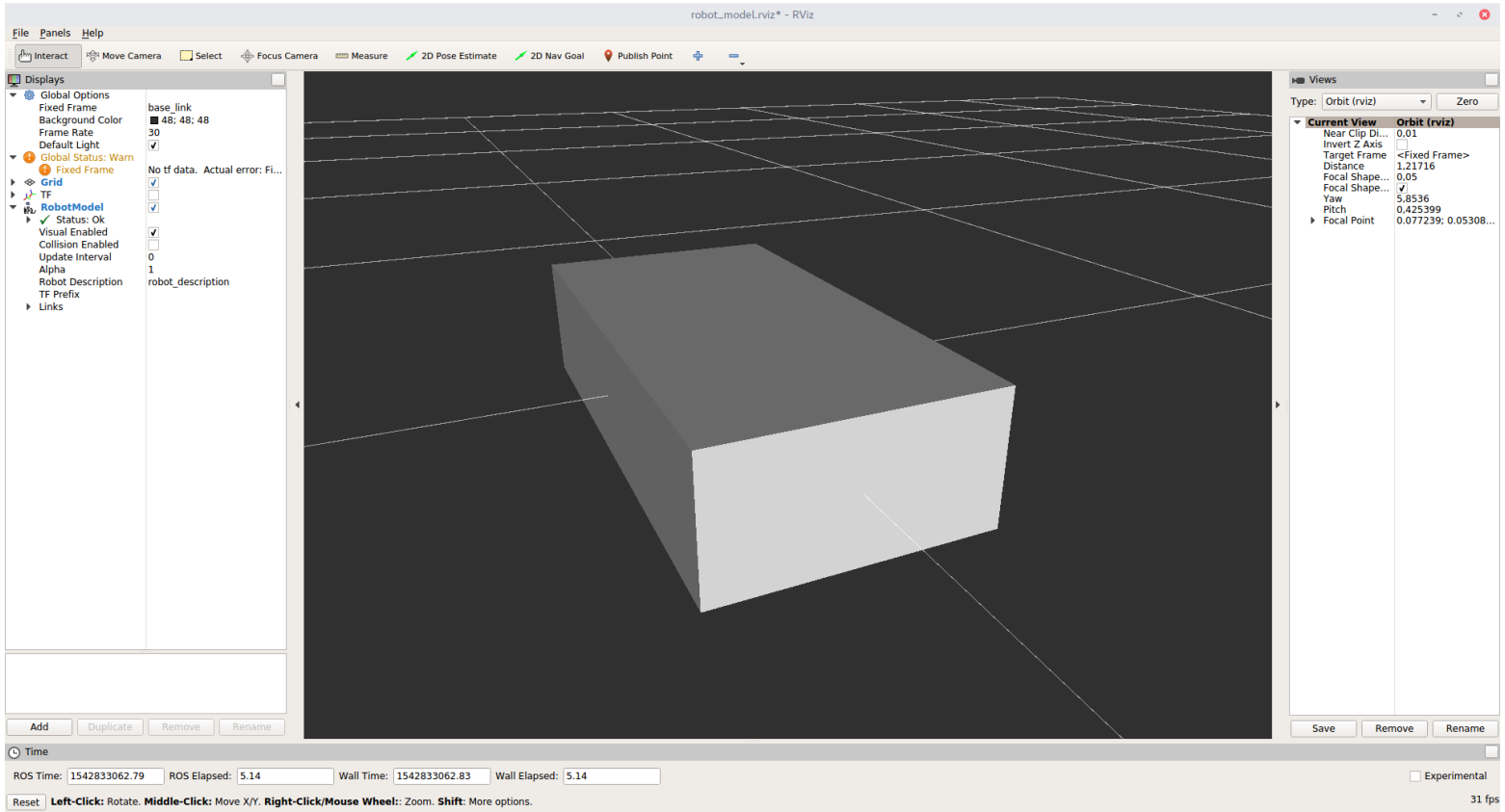
```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro"
name="test_robot">
   <xacro:include filename="body.xacro"/>
   <xacro:body/>
</robot>
```

body.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
   <xacro:macro name="body">
      <link name="base_link">
         <visual>
            <geometry>
               <box size="0.8 0.4 0.2"/>
            </geometry>
            <material name="white">
               <color rgba="1 1 1 1"/>
            </material>
         </visual>
      </link>
   </xacro:macro>
</robot>
```

1. Создание корпуса

```
user@ros: ~/ros/src/roboschool2018/test_robot/urdf/$ roslaunch urdf_viz
xacro.launch filename:=test_car.xacro
```



2. Создание колеса

test_robot.xacro

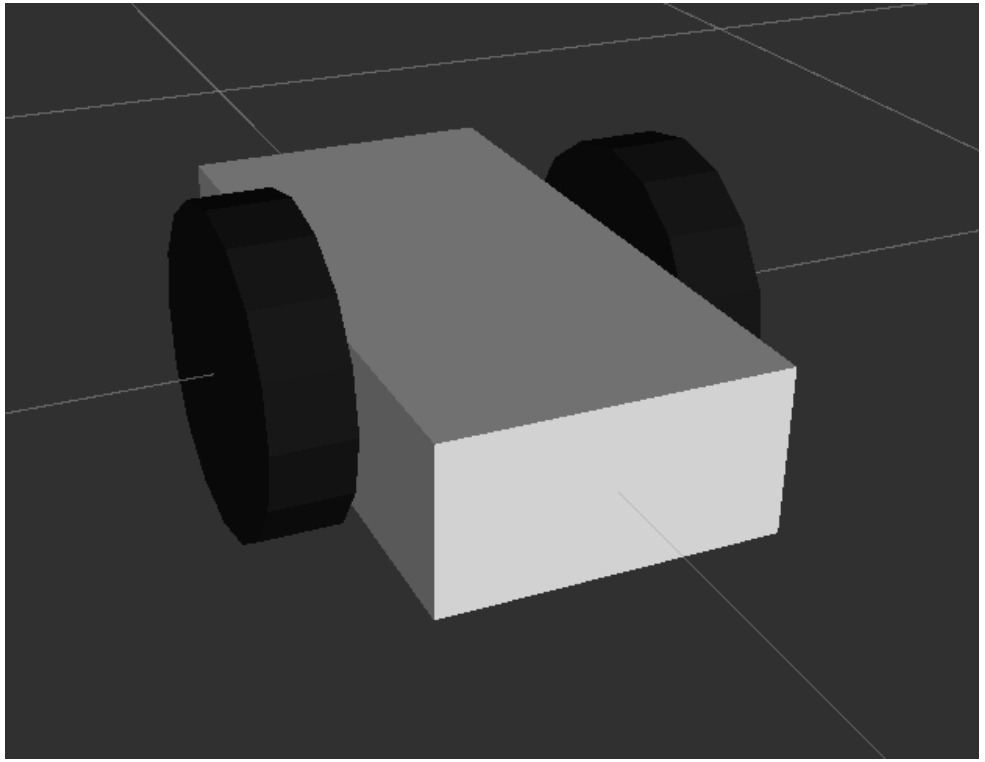
```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro"
name="test_robot">
   <xacro:include filename="body.xacro"/>
   <xacro:include filename="wheel.xacro"/>
   <xacro:body/>
   <xacro:wheel name="wheel_l" side="left"
parent="base_link"/>
   <xacro:wheel name="wheel_r" side="right"
parent="base_link"/>
</robot>
```

wheel.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
   <xacro:macro name="wheel" params="parent side name">
      <xacro:if value="${side == 'left'}">
         <xacro:property name="x_sign" value="1" />
      </xacro:if>
      <xacro:if value="${side == 'right'}">
         <xacro:property name="x_sign" value="-1" />
      </xacro:if>
      <link name="${name}">
         <visual>
            <geometry>
               <cylinder radius="0.2" length="0.1"/>
            </geometry>
            <material name="black">
               <color rgba="0.1 0.1 0.1 1"/>
            </material>
         </visual>
      </link>
      <joint name="${name}_joint" type="continuous">
         <parent link="${parent}"/>
         <child  link="${name}"/>
         <origin xyz="0 ${x_sign*0.25} 0" rpy="“1.57 0 0"/>
         <axis xyz="0 0 1"/>
      </joint>
   </xacro:macro>
</robot>
```

2. Создание колеса

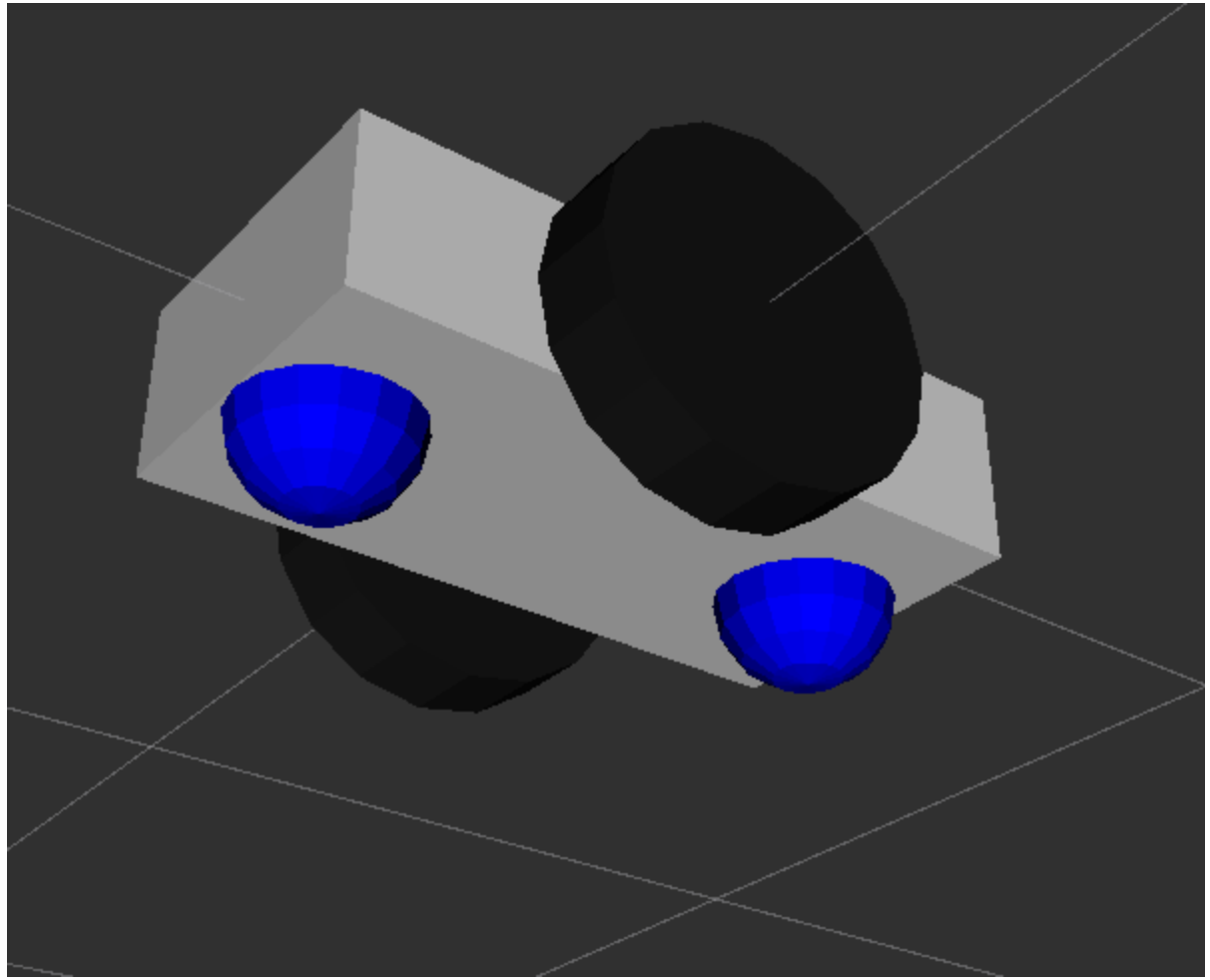
```
user@ros: ~/ros/src/roboschool2018/test_robot/urdf/$ roslaunch urdf_viz
xacro.launch filename:=test_car.xacro
```



3. Костыли

body.xacro

```
<link name="base_link">
   ...
   <visual>
      <origin xyz="0.3 0 -0.1" rpy="0 0 0"/>
      <geometry>
         <sphere radius="0.1"/>
      </geometry>
      <material name="blue">
         <color rgba="0 0 1 1"/>
      </material>
   </visual>
   <visual>
      <origin xyz="-0.3 0 -0.1" rpy="0 0 0"/>
      <geometry>
         <sphere radius="0.1"/>
      </geometry>
      <material name="blue">
         <color rgba="0 0 1 1"/>
      </material>
   </visual>
</link>
```



4. Добавление коллизий

body.xacro

```
...
<link name="base_link">
   ...
   <collision>
      <geometry>
         <box size="0.8 0.4 0.2"/>
      </geometry>
   </collision>
   <collision>
      <origin xyz="0.3 0 -0.1" rpy="0 0 0"/>
      <geometry>
         <sphere radius="0.1"/>
      </geometry>
   </collision>
   <collision>
      <origin xyz="-0.3 0 -0.1" rpy="0 0 0"/>
      <geometry>
         <sphere radius="0.1"/>
      </geometry>
   </collision>
</link>
...
```

wheel.xacro

```
...
<link name=«${name}»
  ...
 <collision>
    <geometry>
       <cylinder radius="0.2" length="0.1"/>
    </geometry>
  </collision>
</link>
...
```

4. Добавление массы и инерции

body.xacro

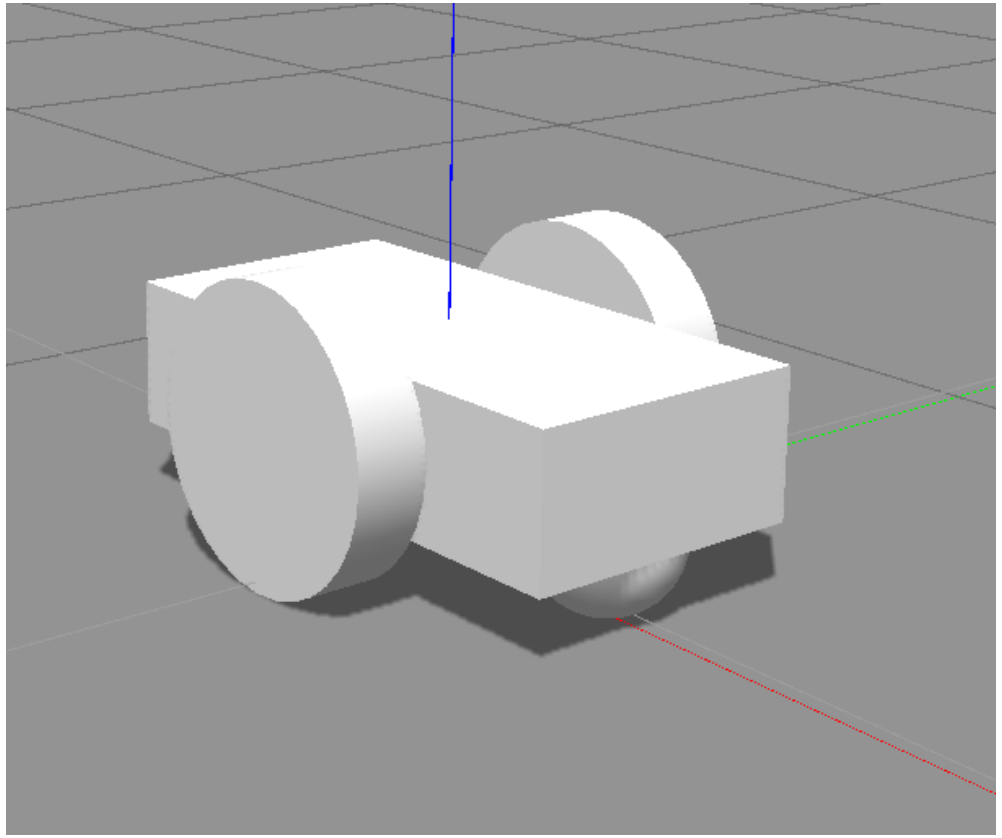
```
...
<link name="base_link">
   ...
   <inertial>
         <mass value="1"/>
         <inertia ixx="0.057" ixv="0.000" ixz="0.000"
                              ivv="0.067" ivz="0.000"
                                          izz="0.017"/>

   </inertial>
</link>
...
```

wheel.xacro

```
...
<link name="${name}">
   ...
   <inertial>
         <mass value="0.3"/>
         <inertia ixx="0.003" ixv="0.000" ixz="0.000"
                              ivv="0.003" ivz="0.000"
                                          izz="0.030"/>

   </inertial>
</link>
...
```

5. Запуск в Gazebo

```
user@ros: ~/ros $ roslaunch test_robot gazebo.launch
```



Q: Где ЦВЕТ?

A: Настройки материалов Gazebo отличаются от стандартных URDF

Q: Что же делать?

A:

- Добавить специфичные для Gazebo теги

```
<gazebo reference="base_link">
    <material>Gazebo/Grey</material>
</gazebo>
```

- Использовать модель Collada (.dae) с материалами или текстурами!

6. Плагин для модели

1. Наследовать класс gazebo::ModelPlugin
2. При инициализации получить указатели на джоинты из модели
3. Инициализировать ROS
4. ???
5. PROFIT

Learn more:

https://github.com/Garrus007/roboschool2018/tree/master/test_robot

http://gazebosim.org/tutorials?tut=plugins_model

7. Сборка плагина

```
user@ros: ~/ros $ catkin_make
...
...
[100%] Linking CXX shared library /home/humanoid/ros-workspace/devel/lib/libtest_robot_plugin.so
[100%] Built target test_robot_plugin
```

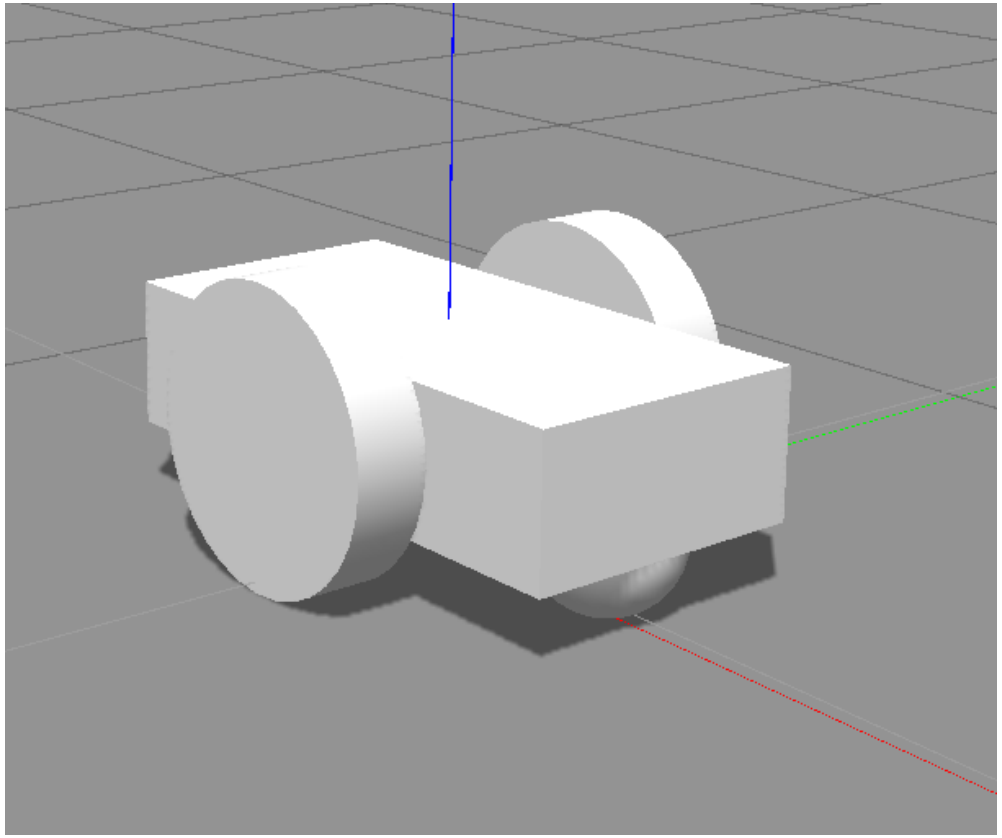
Это надо указать в URDF

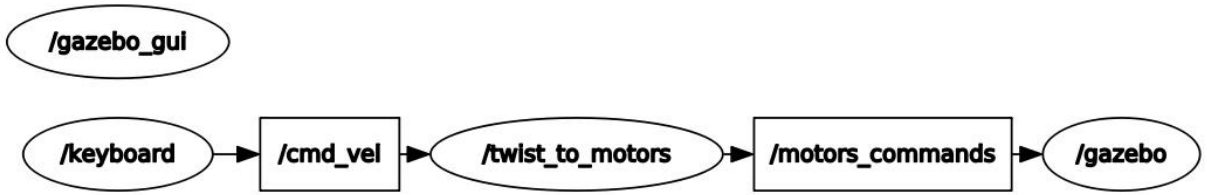
8. Подключение плагина в URDF

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="test_robot">
    ...
    <gazebo>
        <plugin name="test_robot_plugin" filename="libtest_robot_plugin.so"/>
    </gazebo>
</robot>
```

9. Запуск

```
user@ros: ~/ros $ roslaunch test_robot keyboard.launch
```





Спасибо за внимание